

# Intelligent publish-and-subscribe mechanisms for airspace information management

Robert Filman

RIACS

NASA Ames Research Center

[rfilman@mail.arc.nasa.gov](mailto:rfilman@mail.arc.nasa.gov)

650-604-1250

Steve Fonseca

SAIC



# Air System Peril


- Joint Planning and Development Office (JPDO), Next Generation Air Transportation System Integrated Plan (Draft 3, Oct. 2004):

- “The U.S. air transportation system as we know it is in peril. The demand for air transportation is outpacing our ability to increase capacity in our airports. Operating and maintenance costs of the air traffic system are outpacing revenues and conventional air carriers are in serious financial jeopardy. The terrible events of September 11, 2001 radically altered our country and they exposed a new impediment to the future of the air transportation industry. New security requirements are significantly impacting costs and the ability to efficiently move people and cargo. In addition, the growth in air transportation has provoked new community concerns over aircraft noise, pollution and congestion that affect our ability to respond adequately or rapidly enough to our changing world.”



# JPDO Strategies

- Developing the airport infrastructure
- Embedding security measures throughout the air transportation system – from curb to curb.
- Devising alternative concepts of airspace and airport
- Providing each traveler and operator in the system with the specific situational awareness they need.
- Comprehensive and proactive safety management
- Minimize environmental impact
- Reducing weather impacts on air travel
- Developing and employing uniform standards, procedures, and air and space transportation policies worldwide



# System-wide Information Management (SWIM)

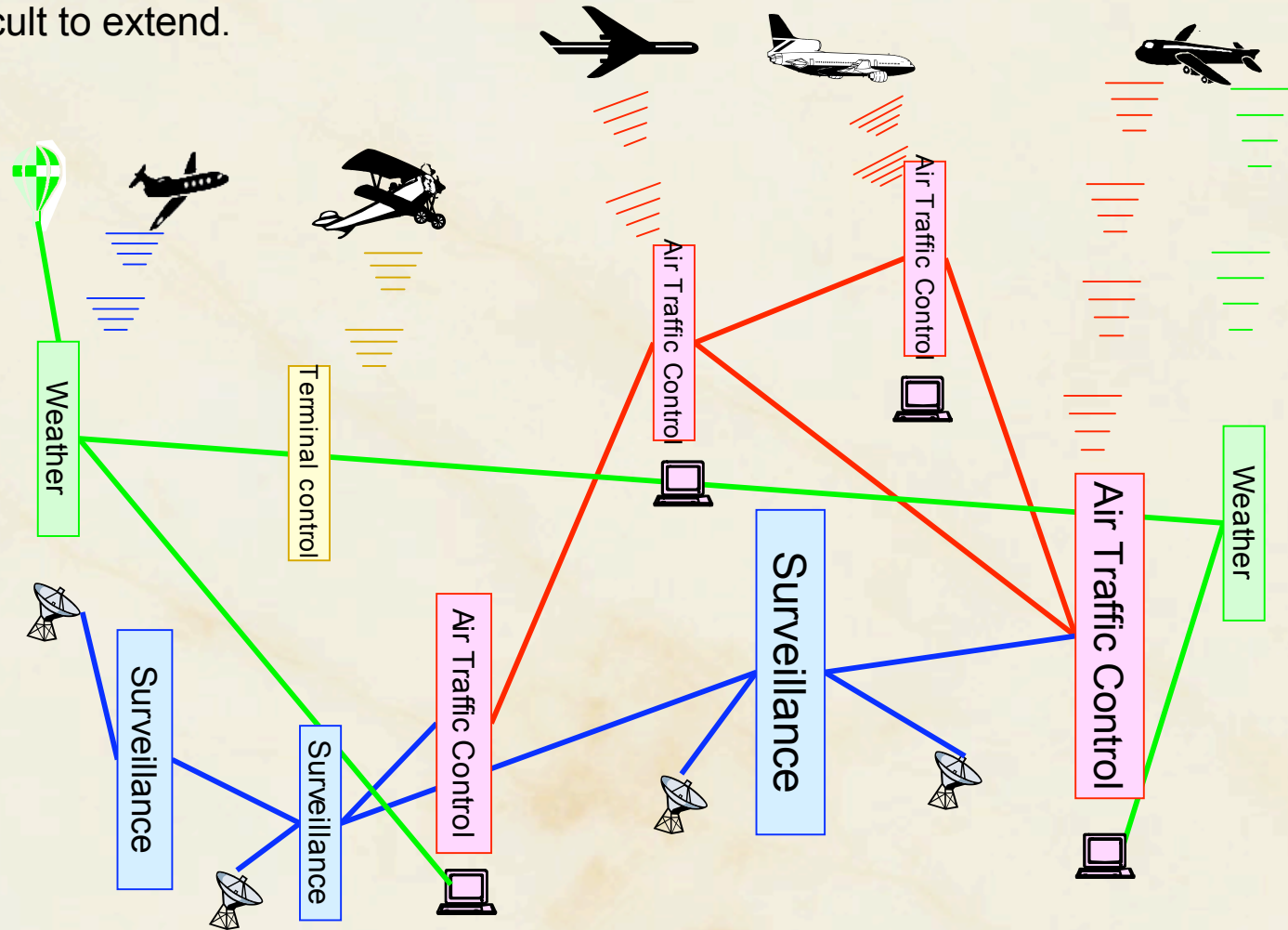
- Treat the National Airspace as an aviation system
- Want to have all “appropriate” information flow throughout the system
  - From the information creators
  - To the information consumer

Current and forecast weather, radar summaries, hazardous condition warnings, airport and airspace capacity constraints, temporary flight restrictions, special use airspace schedules, flight information on each flight, first movement of aircraft, position data inflight, touchdown time, gate and parking assignments, electronic navigation, maps, charts, airport guides, notices to airmen, schedules



# Current Airspace Info is Stovepiped

- Current air traffic information is carried by stand-alone networks and processed by stove-pipe applications.
- Information interchange is special-purpose, often human-mediated, and difficult to extend.



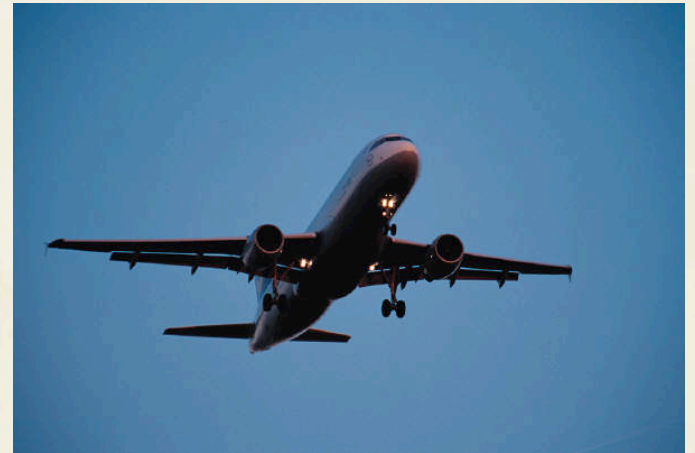
# Motivation

- Achieving system-wide information management has many elements
  - Physical communication
  - Policies and procedures
- This work concentrates on the *abstract communication architecture*
- Application view of the underlying system
  - What communication primitives exist
  - How they interact
  - Achieving ilities:
    - Security
    - Reliability
    - Efficiency
    - Evolvability
    - etc.



# Abstract Communication

- Rigid information architecture
  - Human mediated
- Service-based
  - Procedure call
- Event-based
  - Directed
  - Broadcast
  - Publish and subscribe



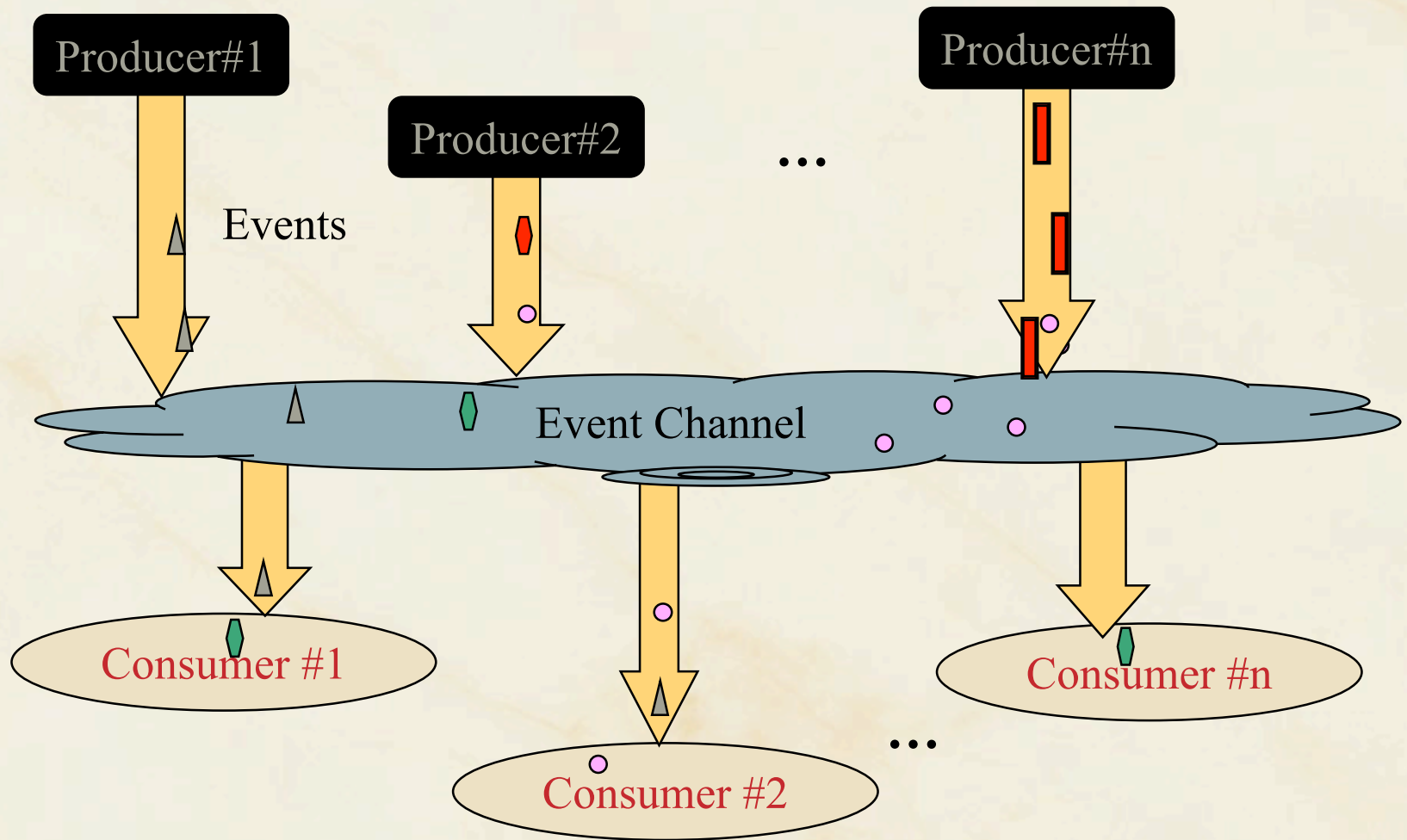


# Publish and Subscribe

- Publishers publish *events*
- Subscribers provide *subscriptions*, which specify which events are interesting
- The underlying system (the *event channel*) arranges to route to each subscriber every event that matches that subscriber's subscription.
  - Modulo policies such as
    - Security
    - Priorities
    - Accounting



# Publish and Subscribe






# Architectural Decisions

- What is an event
- What is a subscription
- Push/pull delivery
  - Subscribers are notified when a subscribed event has occurred
  - Subscribers can poll to see if there are any waiting events
- Archive of events
- How to handle anomalies
  - Failure notification
  - Being offline when an event of interest happens
- Intermediary architecture
- How to enforce policies
- How to minimize the amount of unnecessary communication



# Efficiency

- Precision: What fraction of the events forwarded to a subscriber are actually “interesting”
  - Correctness
  - Timeliness
- Matching efficiency: How much work has to be done by publishers and the event channel to match events to subscribers
- Distribution efficiency: How can messages be routed so as to minimize the number of transmissions



# Efficiency in the National Airspace

## ● Hypothesis:

- There will be many events
- Most events will be of interest to only a small set of subscribers
- There will be a variety of dimensions of locality of interest
- The cost of communication will increasingly continue to exceed the cost of computation



# Subscriptions as languages


- Imposing different rules on recognition mechanisms allows
  - Recognizing different sets
  - Different computational loads





# Language hierarchies (Chomsky hierarchy)

- Task: Define a subset of all strings over a given alphabet
- Type-0 grammars (unrestricted grammars)
  - Turing-machine equivalent
  - Recursively enumerable languages
- Type-1 grammars (context-sensitive grammars)
  - Context-sensitive languages.
  - Recognized by a non-deterministic Turing machine whose tape is bounded by a constant times the length of the input.

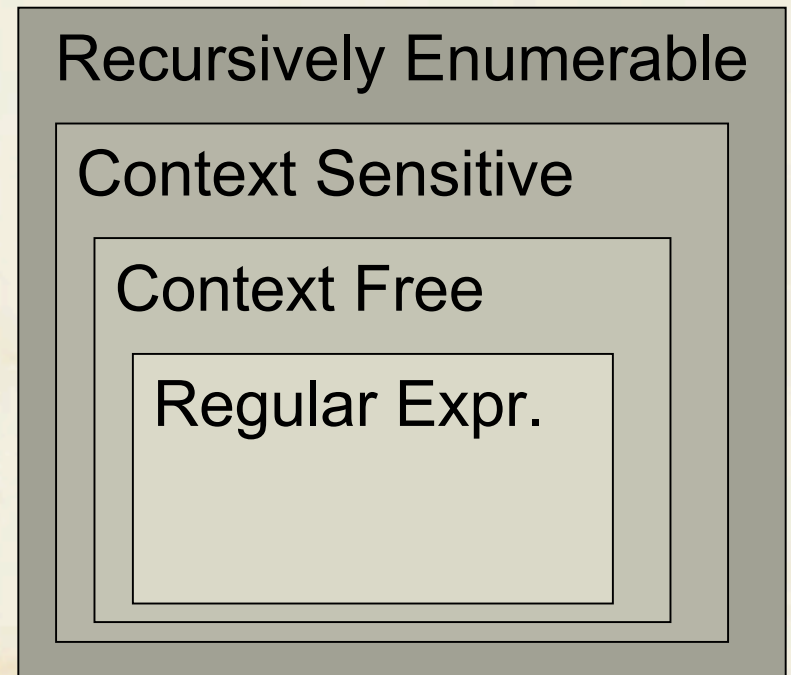


# Chomsky Hierarchy, con't

- Type-2 grammars (context-free grammars)
  - Context-free languages.
  - Recognized by a non-deterministic pushdown automaton
- Type-3 grammars (regular grammars)
  - Regular languages.
  - Recognized by a finite state automaton.
  - Regular expressions.

# Language Hierarchies

- Inclusive:
  - Each category recognizes everything below it
  - Each category includes some things that aren't in language below it
- Increasing computational complexity







# Subscription Language Hierarchies

- Types
- Predicates
- Temporal expressions
- Programs
- Are we recognizing single events or patterns of events?




# What's an event

- Map
  - From hierarchical (perhaps subscripted) names
  - To values (which have their natural interpretations)
- All events may have certain predefined “fields”
  - Time sent
  - Sender
  - Class
  - Etc.
- Represented either generically or by an application-defined structure
- Envelope vs. content



# Type Subscription Languages

- Single events by class
  - “All weather events”
- This is the way most current pub/sub systems work
- Quite efficient at matching efficiency
- Broadcast mechanisms can be used to improve distribution efficiency
- Weak precision
  - Flying into Duluth I’m unlikely to be interested in Tampa weather



# Predicate subscription languages

- Events by predicate
  - “All weather events for Seattle since 5:00”
- Allows
  - Greater precision
  - Weaker matching efficiency
    - But still state-free
  - Weaker distribution efficiency





# Temporal event languages

- Subscriber may be interested not in single events, but in patterns of events over time
  - A sequence of delayed takeoffs from an airport
  - The first login event after five recent login failures using different passwords on the same account
  - The late arrival of a plane carrying a passenger who will be the pilot of a plane still scheduled to take off on time



# Temporal efficiency

- Allows greater precision
- Much weaker matching efficiency
  - The event matcher has to keep some local state representing past events
- Similar distribution efficiency

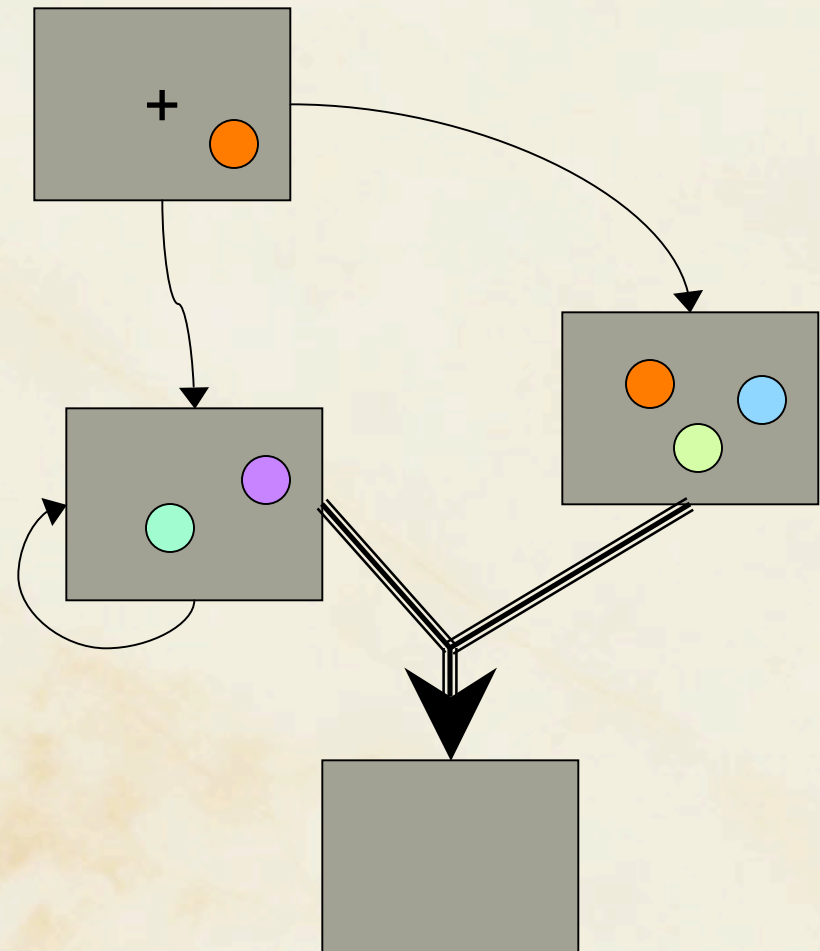


# Temporal language candidates

- Want to be able to express
  - Predicates over single events
  - Temporal relationships
    - Time between events
    - Timeouts
  - What the output should be
- Temporal logic
  - Always, eventually, ....
- Token-based matching automata
  - UML states
  - Petrie nets
  - RETE

# Temporal matching automata

- Nodes
  - Hold tokens
  - May output
- Edges
  - Ordinary edges
  - Join edges
  - May output
- Tokens
  - Hold event data state







# Nodes

- Label
- (Optional) expression defining what output is to be sent to the subscriber when a token enters this node
- One node is the *start* node
  - Which has an infinite supply of blank tokens
- Some nodes are sinks (no output edges)
  - Tokens that arrive at sinks can be recycled



# Token

- Token have a map from symbols to
  - ▣ Values
  - ▣ Maps
- Every token has a color
  - ▣ Color is preserved over ordinary edges
  - ▣ Color over joins devolves to the first arc



# Ordinary edges

- Connect a source node to a destination node
- Label
- Predicate or a time-out
  - The predicate is on an event
- (optional) filter
  - A predicate on an event and a token
  - For time-out edges, a predicate on a token
- (optional) output



# Ordinary Edge Operational Semantics

- Semantically, an ordinary edge is trying to take tokens from its source to its destination.
- A token *passes over* an ordinary edge when
  - And
    - Or
      - The edge has a time-out and “enough” time has elapsed
      - An event arrives which matches the predicate]
    - The edge filter is satisfied

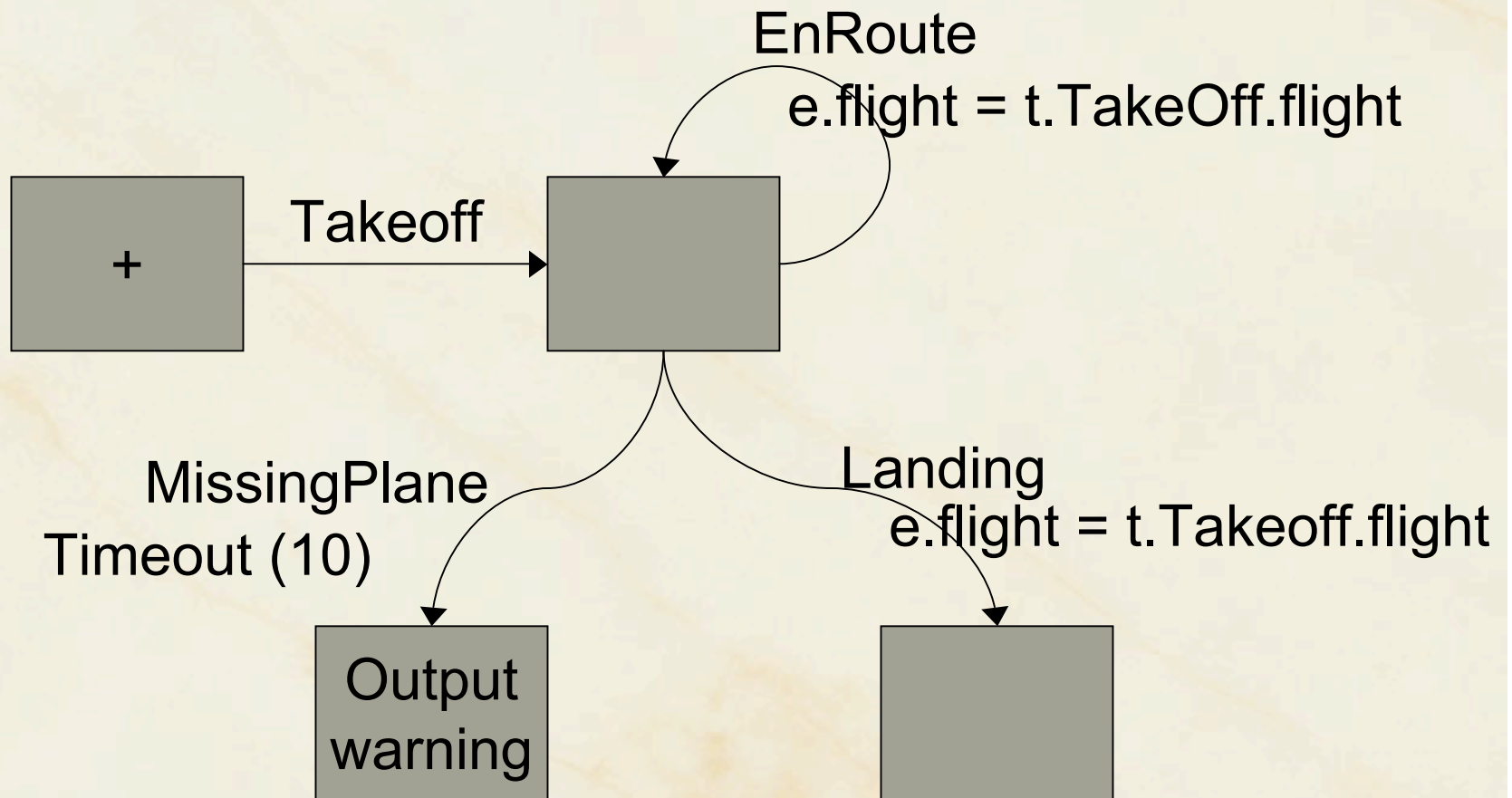




# Tokens over edges

- In passing over an edge, the token is augmented with the map element EdgeLabel -> Token
  - Prior and first uses of that EdgeLabel are also kept in the token

# Ordinary example



- Warn about any enroute flight that hasn't reported in every ten minutes



# Computational efficiency

- Ordinary edges are linearly efficient
  - ▣ Can cull by the predicate
- We're keeping some (but not all) historical state



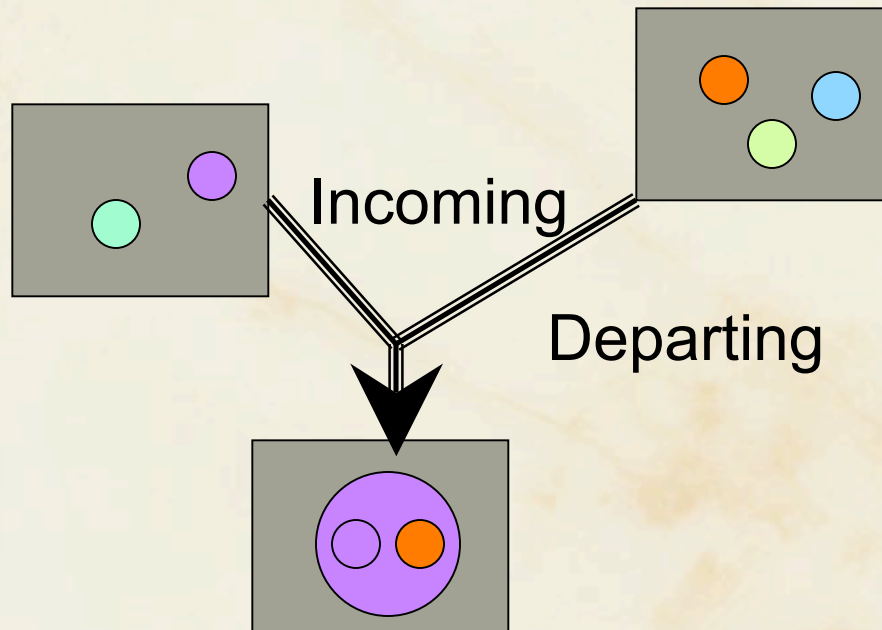
# Join edges

- Have a label on each edge
- Filter in terms of the labels and associated tokens
- Join consumes one token from each source node and builds a token on the destination node with the labels mapped to the corresponding tokens



# Join Example

Incoming.kind = "LateArrival" &  
Departing.kind = "PlannedTakeoff" &  
Incoming.ArrivalTime + 1 hour <  
Departing.DepartureTime &  
Incoming.flight = "AA123" &  
Departing.flight = "United789"





# Join Efficiency

- Joins are potentially inefficient, because (unoptimized) may need to check the cross-product of a several sets of nodes
- Open question about the correspondence of this automata model to various temporal logics



# Subscriptions by program

- Type 0 - Turing equivalent
- Code that runs near the publisher, with state, that can perform arbitrary recognition
- Can have internal state
- Java byte-coded objects
- Lisp S-expressions + environment
- Good semantics for allowing agents to run code on your machine



# Elements of a subscription

- Subscription identity (for management)
- Duration of subscription
- Identity of subscriber
  - For security (want to do security on subscriptions, not messages)
  - For accounting
- Periodicity of desired information
- Freshness requirement for desired information
- How far back in history the subscription extends
- What to do with the results
  - Target
  - Continuation
- Precision, accuracy,
- Source
- Multiplicity





# Hooking it all together

- How does the system arrange that the information from a publisher goes to a subscriber?
- Global knowledge
  - Minimize global knowledge
  - Operationally, need to not perform global consultations for routine events
- Like domain name service (DNS)

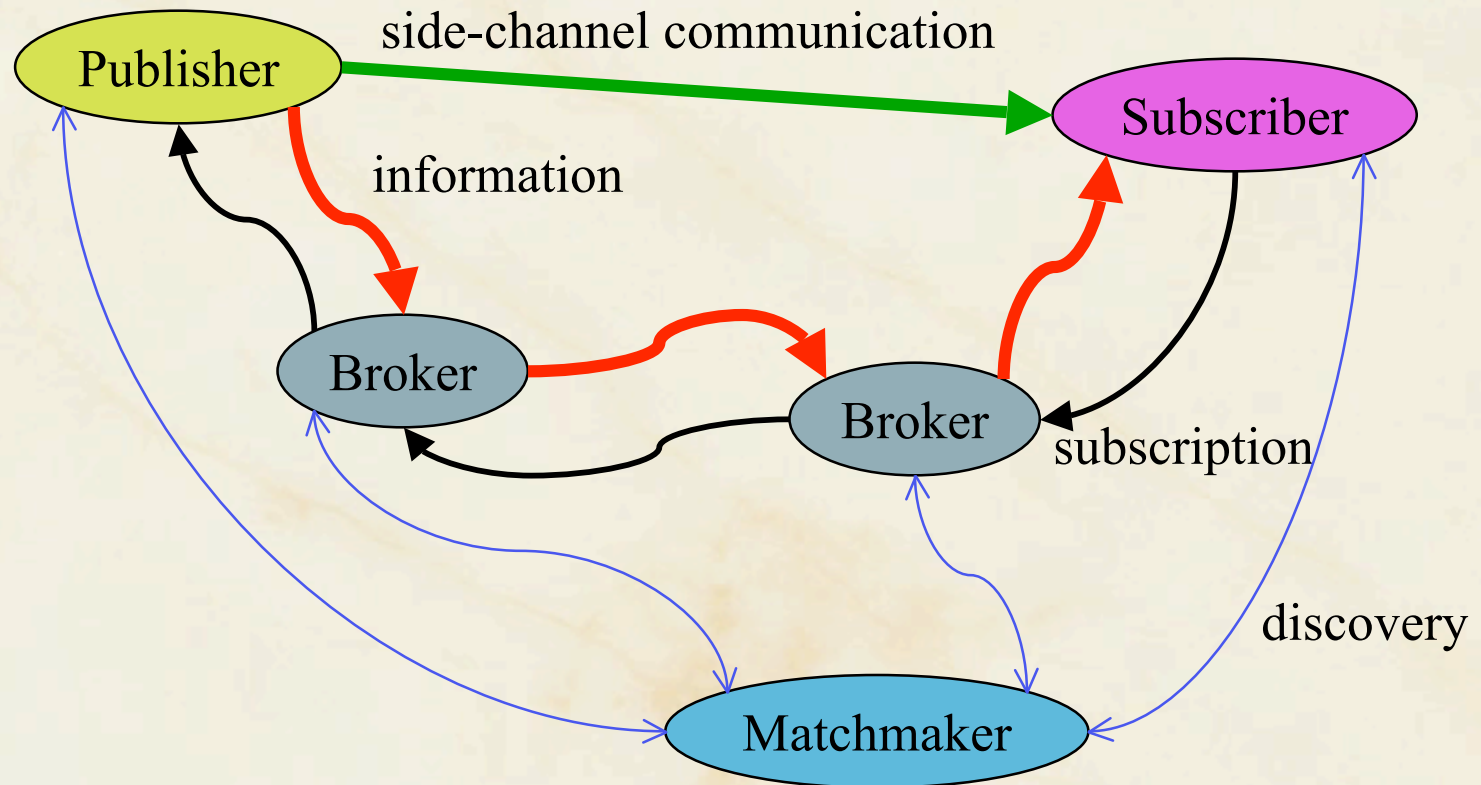
# Architecture

- Matchmaker who is told about
  - Publishers
  - Subscribers can ask the matchmaker who publishes information of interest
- Notification of changes



# Domain Elements

## ● Publishers, Subscribers, Brokers and Matchmakers



Matchmakers are used to find the right brokers



# Matchmaker language

- Publishers describe a set of pairs, <field name, type information>
  - Type can be a set or a unit set
- Subscribers describe their subscriptions in the same terms
- It is possible to express Boolean alternatives about publication topics
  - (A & B) | (C & D)
- By checking for overlap, the matchmaker can tell potential subscribers where they need to send their information



# Matchmaking

- Requires the ability to understand description the class of events generated by a publisher
- Requires the ability to scope the class of events desired by a subscriber

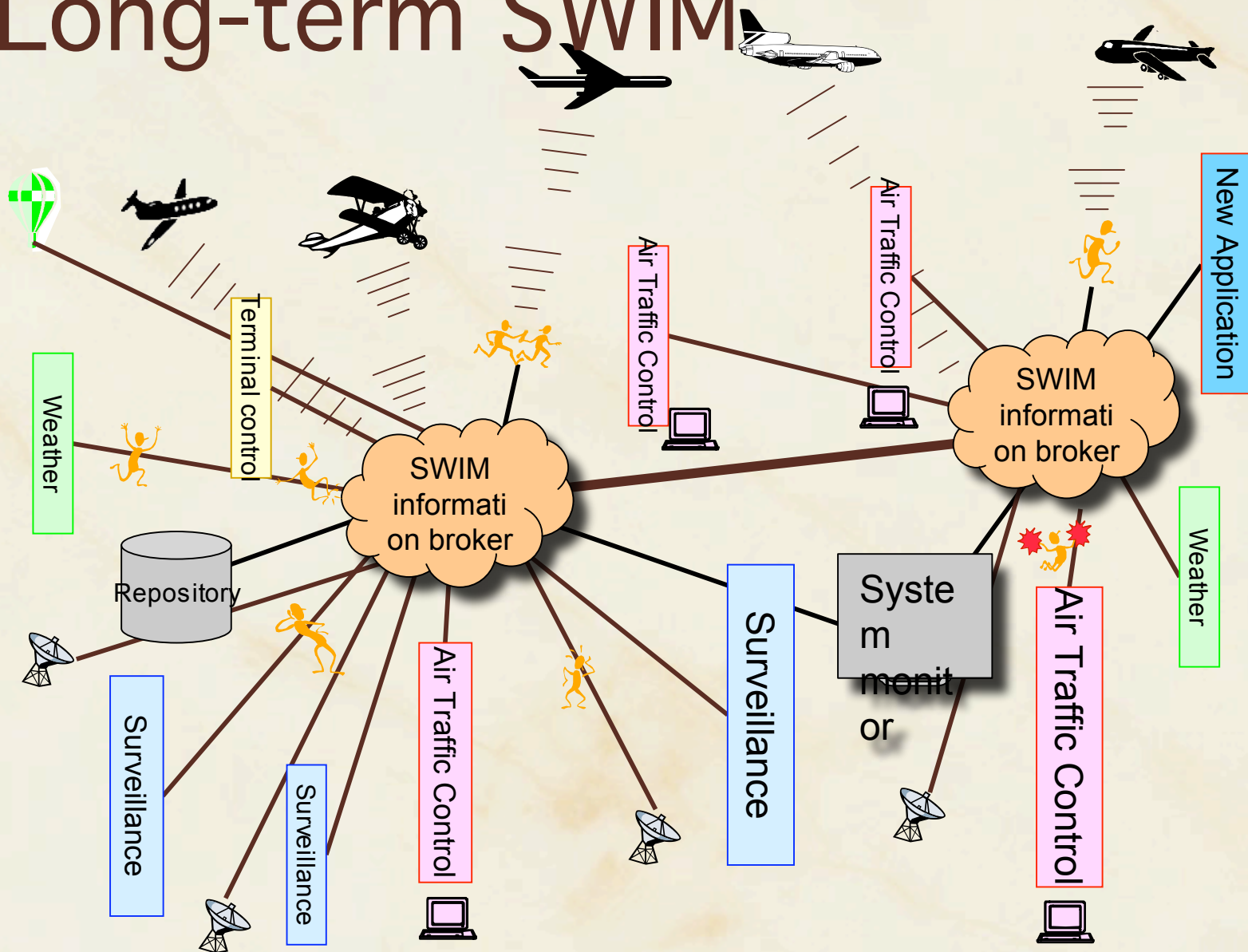




# Filtering (Injectors)

- Allow dynamic message filtering of communications
  - By senders, receivers and intermediaries
  - Facilitates system evolution
    - Enforce new policies
      - Quality of service
      - Reliability
      - Security
    - Convert data between versions
    - Update code between versions
    - Provide manageability

# Long-term SWIM





# Implementation status

- Distributed messaging system
- Several subscription languages implemented
  - (almost)
- Plan on performance testing



